ASSIGNMENT ON OOPS IN PYTHON

BY
SHINO MARY PHILIPOSE

Q.1

FileHandling.py | OopsAssignment.py | Oops.py

```python
# Build a program to manage a university's course catalog. You want to define a base class Course
# that has the following properties:
# course_code: a string representing the course code (e.g., "CS101")
# course_name: a string representing the course name (e.g., "Introduction to Computer Science")
# credit_hours: an integer representing the credit hours for the course (e.g., 3)
# You also want to define two subclasses CoreCourse and ElectiveCourse, which inherit from the Course cla
# CoreCourse should have an additional property required_for_major which is a boolean
# representing whether the course is required for a particular major.
# ElectiveCourse should have an additional property elective_type which is a string representing
# the type of elective (e.g., "general", "technical", "liberal arts").


# Base class for Course
# class Course:
```

Q.1

FileHandling.py OopsAssignment.py Oops.py

```python

#Base class for Course
class Course:  2 usages
    def __init__(self, course_code, course_name, credit_hours):
        self.course_code = course_code  # e.g., "CS101"
        self.course_name = course_name  # e.g., "Introduction to Computer Science"
        self.credit_hours = credit_hours  # e.g., 3

    def __str__(self):
        return f"{self.course_code}: {self.course_name} ({self.credit_hours} credit hours)"

# Subclass for CoreCourse
class CoreCourse(Course):  2 usages
    def __init__(self, course_code, course_name, credit_hours, required_for_major):
        # Call the parent constructor (Course)
        super().__init__(course_code, course_name, credit_hours)
        self.required_for_major = required_for_major  # True or False

```

Q.1

FileHandling.py | OopsAssignment.py | Oops.py

```python
class CoreCourse(Course):  2 usages

    def __str__(self):
        required_status = "Required" if self.required_for_major else "Elective"
        return f"{super().__str__()} - {required_status} for major"

# Subclass for ElectiveCourse
class ElectiveCourse(Course):  2 usages
    def __init__(self, course_code, course_name, credit_hours, elective_type):
        # Call the parent constructor (Course)
        super().__init__(course_code, course_name, credit_hours)
        self.elective_type = elective_type  # e.g., "general", "technical", "liberal arts"

    def __str__(self):
        return f"{super().__str__()} - {self.elective_type} elective"

# Example usage
if __name__ == "__main__":
    # Create some courses
```

Q.1

FileHandling.py | OopsAssignment.py | Oops.py

```python
        return f"{super().__str__()} - {self.elective_type} elective"

# Example usage
if __name__ == "__main__":
    # Create some courses
    course1 = CoreCourse(course_code: "CS101", course_name: "Introduction to Computer Science", credit_hours: 3,
                         required_for_major: True)
    course2 = CoreCourse(course_code: "MATH101", course_name: "Calculus I", credit_hours: 4, required_for_major: False)
    course3 = ElectiveCourse(course_code: "HIST201", course_name: "History of Art", credit_hours: 3,
                             elective_type: "liberal arts")
    course4 = ElectiveCourse(course_code: "CS202", course_name: "Data Structures", credit_hours: 3,
                             elective_type: "technical")

    # Print the course details
    print(course1)
    print(course2)
    print(course3)
    print(course4)
```

Q.1

FileHandling.py | OopsAssignment.py | Oops.py

```python
# Print the course details
print(course1)
print(course2)
print(course3)
print(course4)
```

OopsAssignment

```
C:\Users\sgkur\PycharmProjects\python_d36_ENTRI\.venv\Scripts\python.exe C:\Users\sgkur\PycharmProjects\python_d36
CS101: Introduction to Computer Science (3 credit hours) - Required for major
MATH101: Calculus I (4 credit hours) - Elective for major
HIST201: History of Art (3 credit hours) - liberal arts elective
CS202: Data Structures (3 credit hours) - technical elective

Process finished with exit code 0
```

Q.2

FileHandling.py | OopsAssignment.py | employee.py | main1.py | Oops.py

```python

# 2.Create a Python module named employee that contains a class Employee with attributes name,
# salary and methods get_name() and get_salary(). Write a program to use this module to create an object
# of the Employee class and display its name and salary.

# employee.py module

# class Employee:
#     def __init__(self, name, salary):
#         self.name = name      # Employee's name
#         self.salary = salary  # Employee's salary
#
#     def get_name(self):
#         """Returns the name of the employee."""
#         return self.name
#
#     def get_salary(self):
#         """Returns the salary of the employee."""
#         return self.salary
#
#
```

Q.2

FileHandling.py | OopsAssignment.py | employee.py | main1.py | Oops.py

python_d36_ENTRI C:\Users\sgkur\PycharmProjects\python_d36_ENTRI

```python
#
# # main1.py
#
# # Importing the Employee class from the employee module
# from employee import Employee
#
#
# def main():
#     # Create an Employee object
#     emp1 = Employee("Johny Linz", 50000)
#
#     # Display the name and salary of the employee
#     print(f"Employee Name: {emp1.get_name()}")
#     print(f"Employee Salary: ${emp1.get_salary():,.2f}")
#
#
# if __name__ == "__main__":
#     main()
```

Q.2

FileHandling.py | OopsAssignment.py | employee.py | main1.py | Oops.py

```python
# employee.py module

class Employee:  2 usages
    def __init__(self, name, salary):
        self.name = name      # Employee's name
        self.salary = salary  # Employee's salary

    def get_name(self):  1 usage
        """Returns the name of the employee."""
        return self.name

    def get_salary(self):  1 usage
        """Returns the salary of the employee."""
        return self.salary
```

Q.2

FileHandling.py | OopsAssignment.py | employee.py | main1.py | Oops.py

```python
class Employee:  2 usages
    def get_name(self):  1 usage
        """Returns the name of the employee."""
        return self.name

    def get_salary(self):  1 usage
        """Returns the salary of the employee."""
        return self.salary
```

employee

```
:\Users\sgkur\PycharmProjects\python_d36_ENTRI\.venv\Scripts\python.exe C:\Users\sgkur\PycharmProjects\python_d36_ENTR

rocess finished with exit code 0
```

Q.2

FileHandling.py | OopsAssignment.py | employee.py | main1.py | Oops.py

```python
# main1.py

# Importing the Employee class from the employee module
from employee import Employee


def main():  1 usage
    # Create an Employee object
    emp1 = Employee( name: "Johny Linz",  salary: 50000)

    # Display the name and salary of the employee
    print(f"Employee Name: {emp1.get_name()}")
    print(f"Employee Salary: ${emp1.get_salary():,.2f}")


if __name__ == "__main__":
    main()
```

Q.2

FileHandling.py | OopsAssignment.py | employee.py | main1.py | Oops.py

```python
    print(f"Employee Salary: ${emp1.get_salary():,.2f}")


if __name__ == "__main__":
    main()
```

Run main1

```
C:\Users\sgkur\PycharmProjects\python_d36_ENTRI\.venv\Scripts\python.exe C:\Users\sgkur\PycharmProjects\python_d36_ENTRI\main1.py
Employee Name: Johny Linz
Employee Salary: $50,000.00

Process finished with exit code 0
```